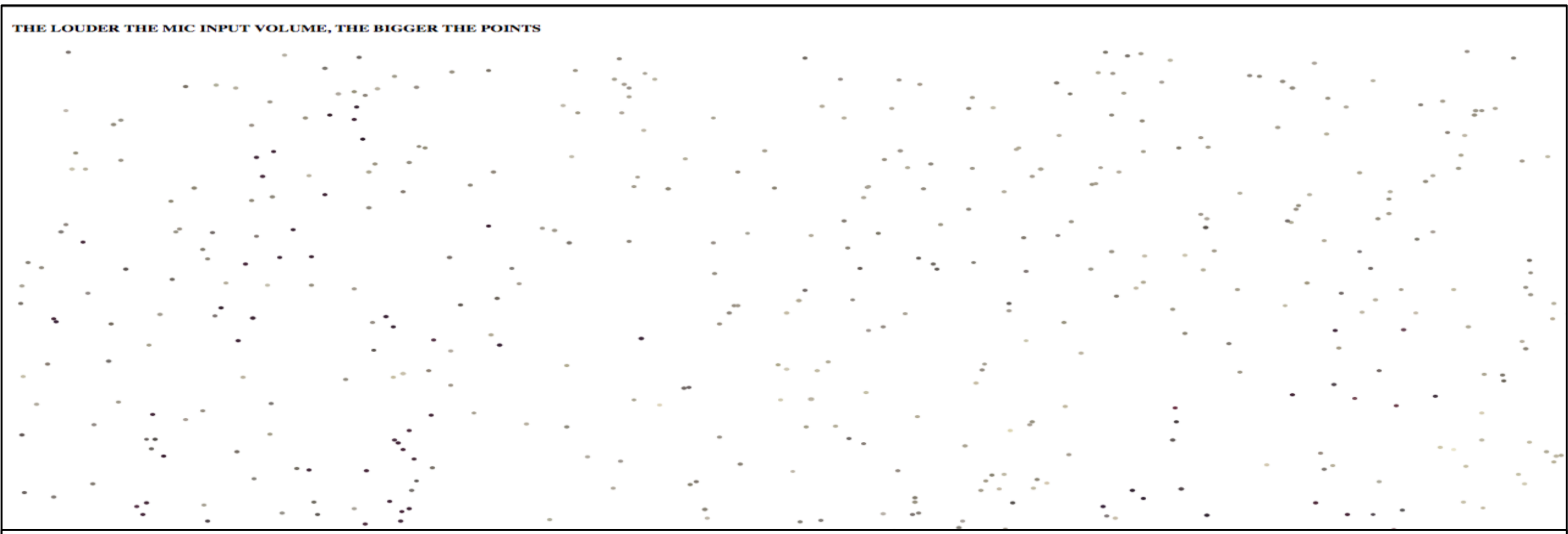
Francisco Samayoa
Adam Tindale
Atelier I: Discovery
September 17, 2018

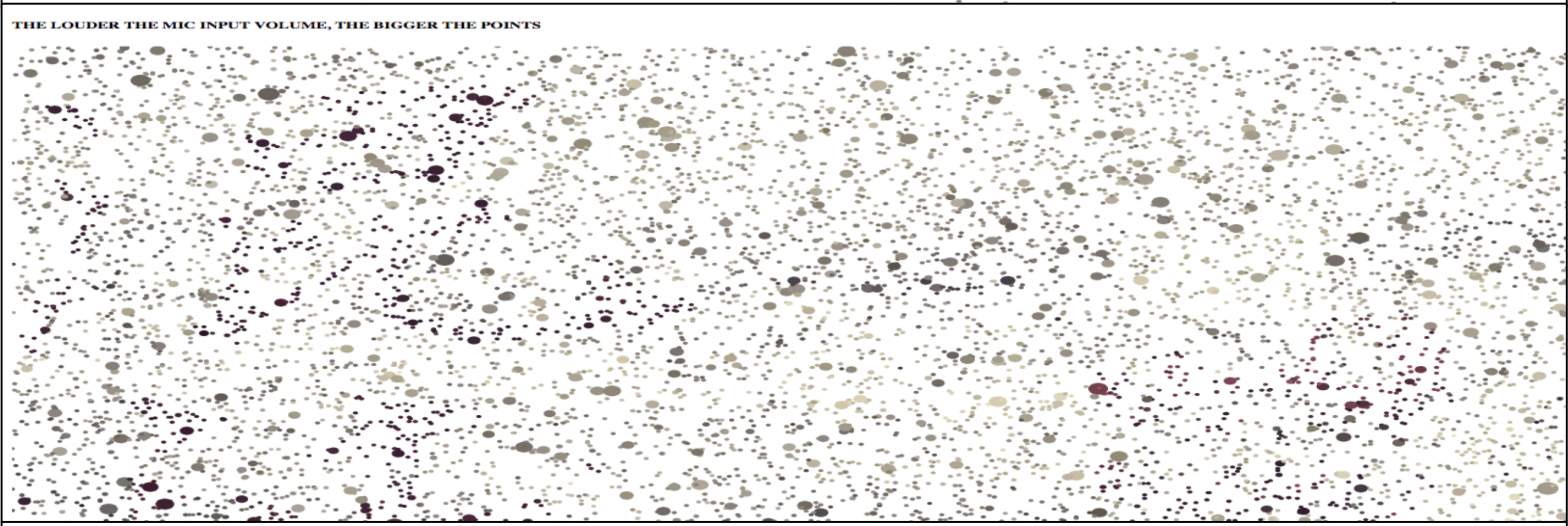
Hello World

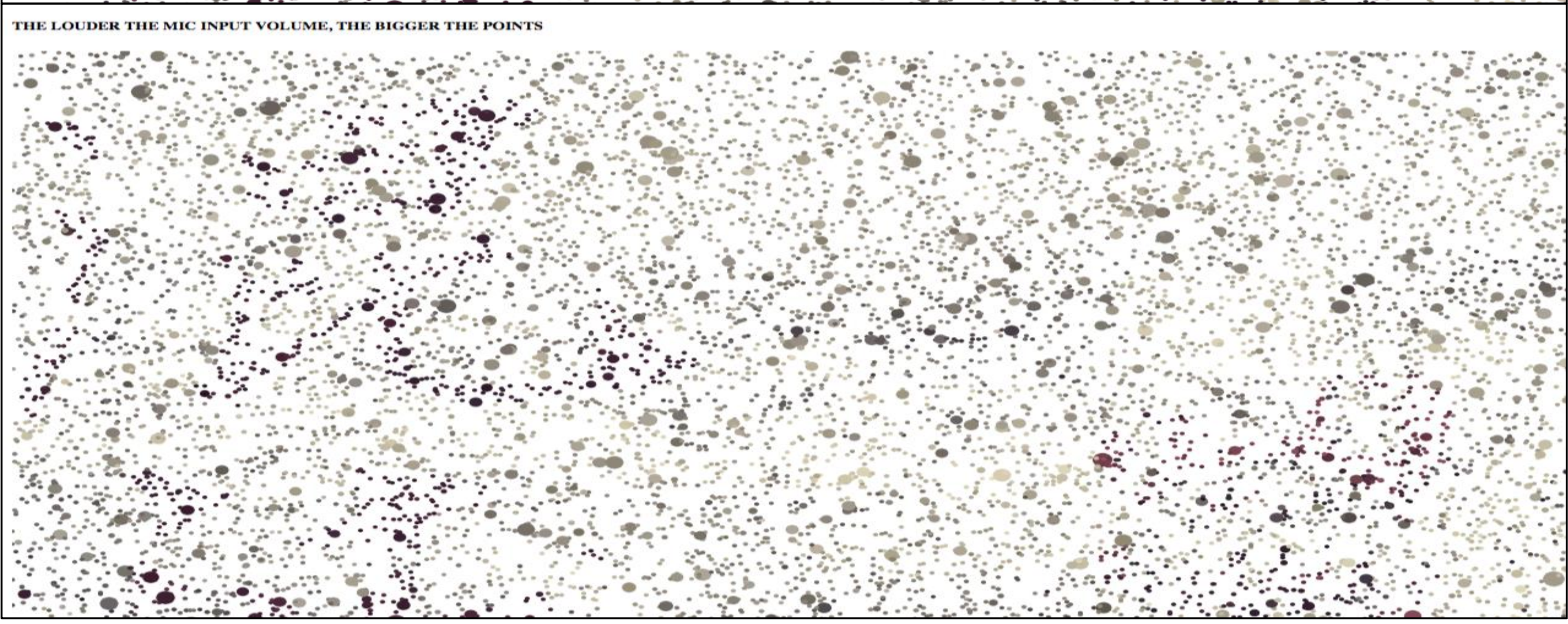
https://github.com/avongarde/Atelier

Sketch One

```
// references mic input example & pointillism example
// https://p5js.org/examples/image-pointillism.html by Dan Shiffman
// https://p5js.org/examples/sound-mic-input.html

var mic;
var img;
var smallPoint, largePoint;

function preload() {
  img = loadImage("image.jpg");
}

function setup() {
  createCanvas(window.innerWidth, window.innerHeight);

  smallPoint = 4;
  largePoint = 40;
  imageMode(CENTER);
  noStroke();
  background(255);
  img.loadPixels();

  mic = new p5.AudioIn();
  // By default, it does not .connect() (to the computer speakers)
  mic.start();
}

function draw() {
  // Get the overall volume (between 0 and 1.0)
  var vol = mic.getLevel();

  var pointillize = map(vol, 0, 1, smallPoint, largePoint);
  console.log(pointillize);

  var x = floor(random(img.width));
  var y = floor(random(img.height));
  var pix = img.get(x, y);
  fill(pix, 128);
  ellipse(x, y, pointillize, pointillize);
}
```


THE LOUDER THE MIC INPUT VOLUME, THE BIGGER THE POINTS


THE LOUDER THE MIC INPUT VOLUME, THE BIGGER THE POINTS

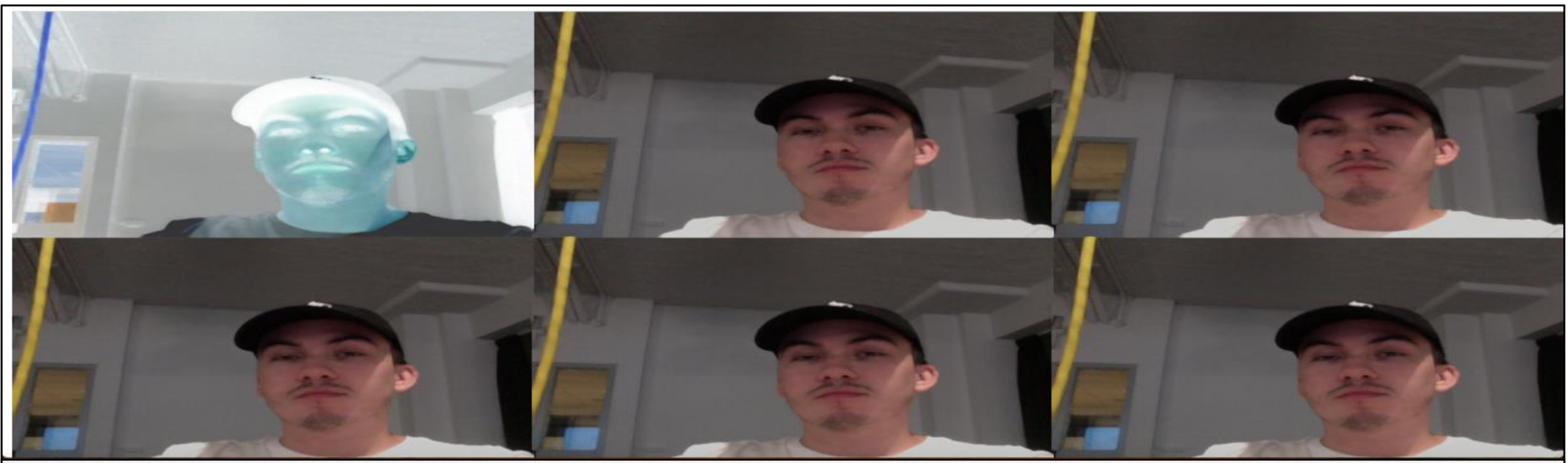
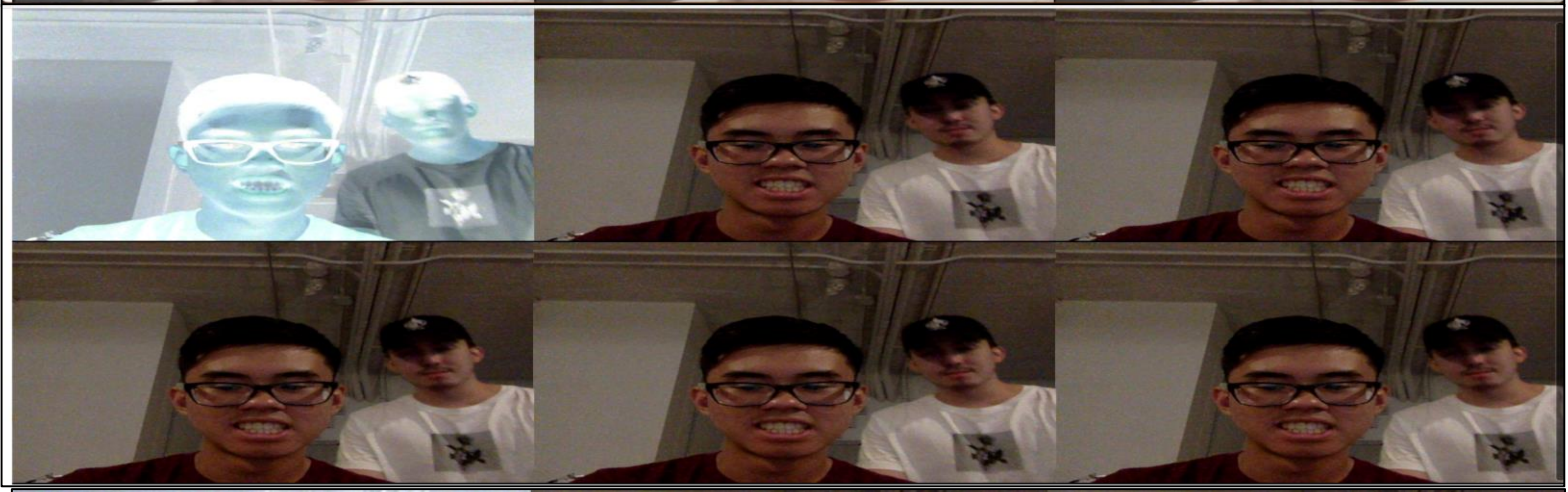

THE LOUDER THE MIC INPUT VOLUME, THE BIGGER THE POINTS

For this sketch, I expanded on an example of image pointillism by Dan Shiffman, a mic input example from the p5 reference page, and the sketch example we began in class. I built upon the idea of using a value to increase the size of the points for the image loading. In this instance I used the volume from the mic input instead of the mouseX from the original example. I also read up on the *img.get(x, y);* aspect in the draw function as I was unsure what it's purpose was. As it turns out, it reads the color of any pixel or grabs a section of an image.

The result was that it worked, although not to my liking. It was quite slow loading the points, so the main change would have to be adjusting the speed at which the points appear. Given more time, I would like the mic input volume to be able to exponentially change the speed as well as the size. It might've also been my choice of image since it was high resolution jpeg.

<u>Sketch 2</u>

```
// references video input example
// https://p5js.org/examples/dom-video-capture.html
var capture;

function setup() {
  createCanvas(window.innerWidth, window.innerHeight);
  capture = createCapture(VIDEO);
  capture.size(480, 320);
  capture.hide();
}

function draw() {
  background(255);
  push();
  image(capture, 0, 0);
  filter('INVERT');
  pop();
  image(capture, width/3, 0);
  // filter('OPAQUE');
  image(capture, width/3 * 2, 0);
  // filter('POSTERIZE');
  image(capture, 0, height/2);
  // filter('GRAY');
  image(capture, width/3, height/2);
  // filter('ERODE');
  image(capture, width/3 * 2, height/2);
  filter('DILATE');
}
```

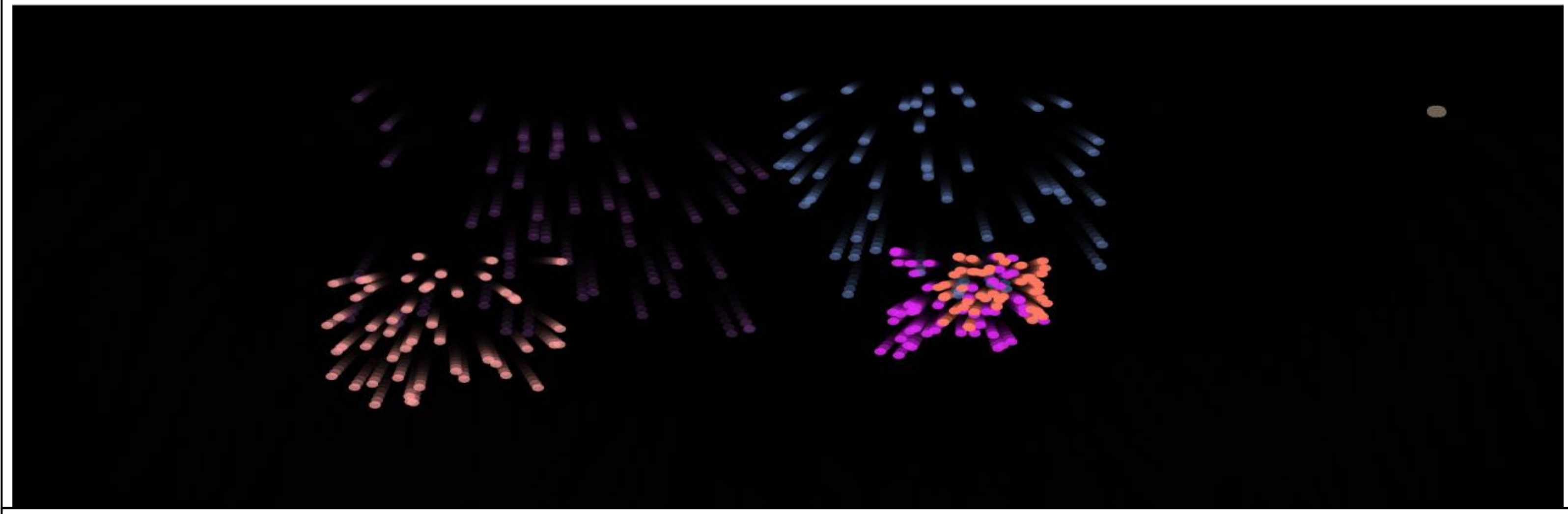
For this sketch, I expanded on the video input example from the p5 reference page. I built upon the idea of using the web camera to capture the viewer, and then present the viewer with six different filters, similar to computers in the Apple Store. I divided the capture screens into six segments, while hiding the initial, unfiltered capture screen outside of the canvas. As shown above, the capture screens were present and functioning.

By referencing the filter page, each segment was supposed to contain a filter variation. At first, the problem was the filter only affected the previous screen. I soon realized that the further down the filter line was, the more screens it affected. It can't affect a single screen, as far as I know. I tried using the *push() and pop()* functions but they failed. Therefore, I would like to find a solution to this if I was given more time. As well as simplifying the code and adding a for loop for creating each screen.

Sketch 3

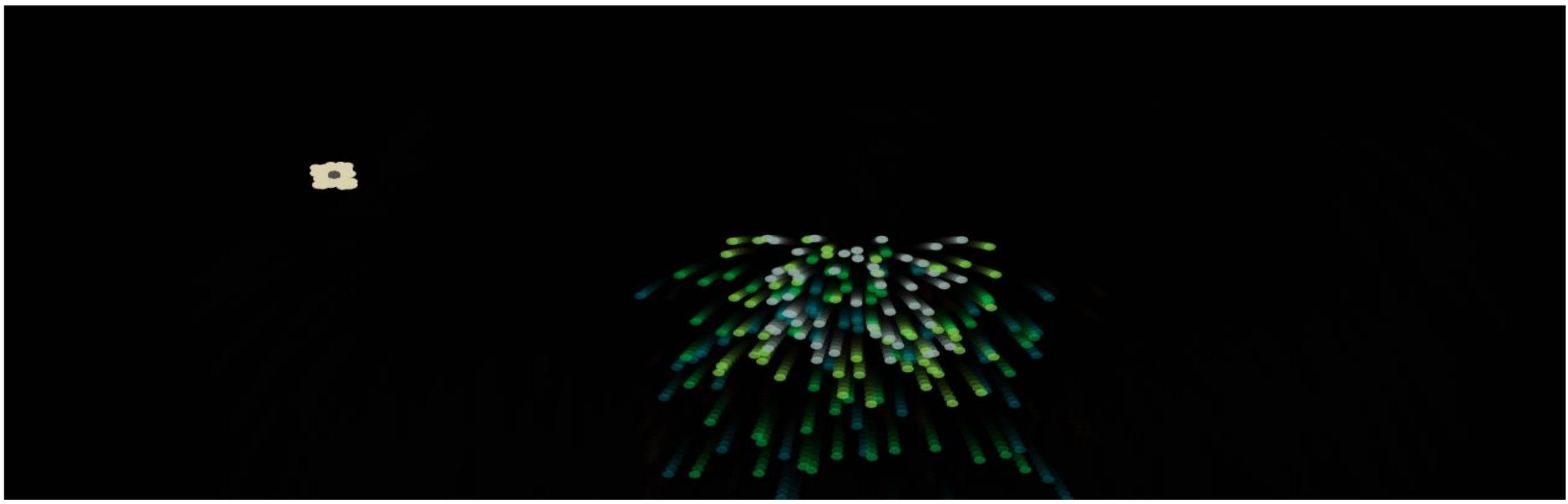
```
var circles = [];

function setup(){
					createCanvas(window.innerWidth, window.innerHeight);
}

function draw(){
					background(0, 0, 0, 100);
					for (var i = 0; i < circles.length; i++){
						noStroke();
						circles[i].lifespan += 0.1;
						circles[i].alpha -= circles[i].lifespan;
						fill(circles[i].r, circles[i].g, circles[i].b, circles[i].alpha);
						ellipse(circles[i].x, circles[i].y, 10, 10);
						circles[i].y += circles[i].ySpeed;
						circles[i].x += circles[i].xSpeed;
						circles[i].ySpeed += 0.1;
					}
}

function mouseClicked(){
					var r = random(255);
					var g = random(255);
					var b = random(255);

					for(var i = 0; i < 50; i++){
						var particle = {
							x: mouseX,
							y: mouseY,
							xSpeed: random(-3, 3),
							ySpeed: random(-3, 3),
							lifespan: 0,
							alpha: 255,
							r: r,
							g: g,
							b: b
						}
					circles.push(particle);
					}
}
```

CLICK AROUND THE CANVAS



CLICK AROUND THE CANVAS

CLICK AROUND THE CANVAS



For this original sketch I wanted to create a display of fireworks with each mouse click. I remember making a similar sketch in last year's Creative Art & Code class and utilized what I remembered. I built upon the basic idea of using a for loop and an array to create ellipses wherever I clicked. The challenge was the direction at which they fell and their random colours.

What worked in the sketch was the realism. The lifespan (the time each particle is alive) is used to lower the alpha overtime, causing them to fade and creating a realistic firework effect. The rgb values had to be declared outside the for loop in order for each mouse click to have its own set colour. *Circles.push(particle)* then adds the particle object to the array. I am content with how the code turned out, but if there was one thing I had to expand it would be adding sound effects.

<u>Sketch 4</u>

```
// references sketch 3 & Pan Sound example
// https://p5js.org/examples/sound-pan-sound.html

var sound;
var circles = [];

function preload(){
  sound = loadSound('sound.mp3');
  console.log(sound);
}

function setup(){
  var cnv = createCanvas(window.innerWidth, window.innerHeight);
  cnv.mouseClicked(togglePlay);
  console.log(togglePlay);
  sound.amp(0.2);
}

function draw(){
  background(0);
  for (var i = 0; i < circles.length; i++){
    noStroke();
    circles[i].lifespan += 0.1;
    circles[i].alpha -= circles[i].lifespan;
    fill(circles[i].r, circles[i].g, circles[i].b, circles[i].alpha);
    ellipse(circles[i].x, circles[i].y, 10, 10);
    circles[i].y += circles[i].ySpeed;
    circles[i].x += circles[i].xSpeed;
    circles[i].ySpeed += 0.1;
  }
}

function mouseMoved(){
  var r = random(255);
  var g = random(255);
  var b = random(255);
  var panning = map(mouseX, 0., width, -1.0, 1.0);
  sound.pan(panning);
  // sound.play();

  for(var i = 0; i < 50; i++){
    var particle = {
      x: mouseX,
      y: mouseY,
      xSpeed: random(-3, 3),
      ySpeed: random(-3, 3),
      lifespan: 0,
      alpha: 100,
      r: r,
      g: g,
      b: b
    }
  circles.push(particle);
  }
  return false;
}

// fade sound if mouse is over canvas
function togglePlay() {
  if (sound.isPlaying()) {
    sound.pause();
  } else {
    sound.loop();
  }
}
```

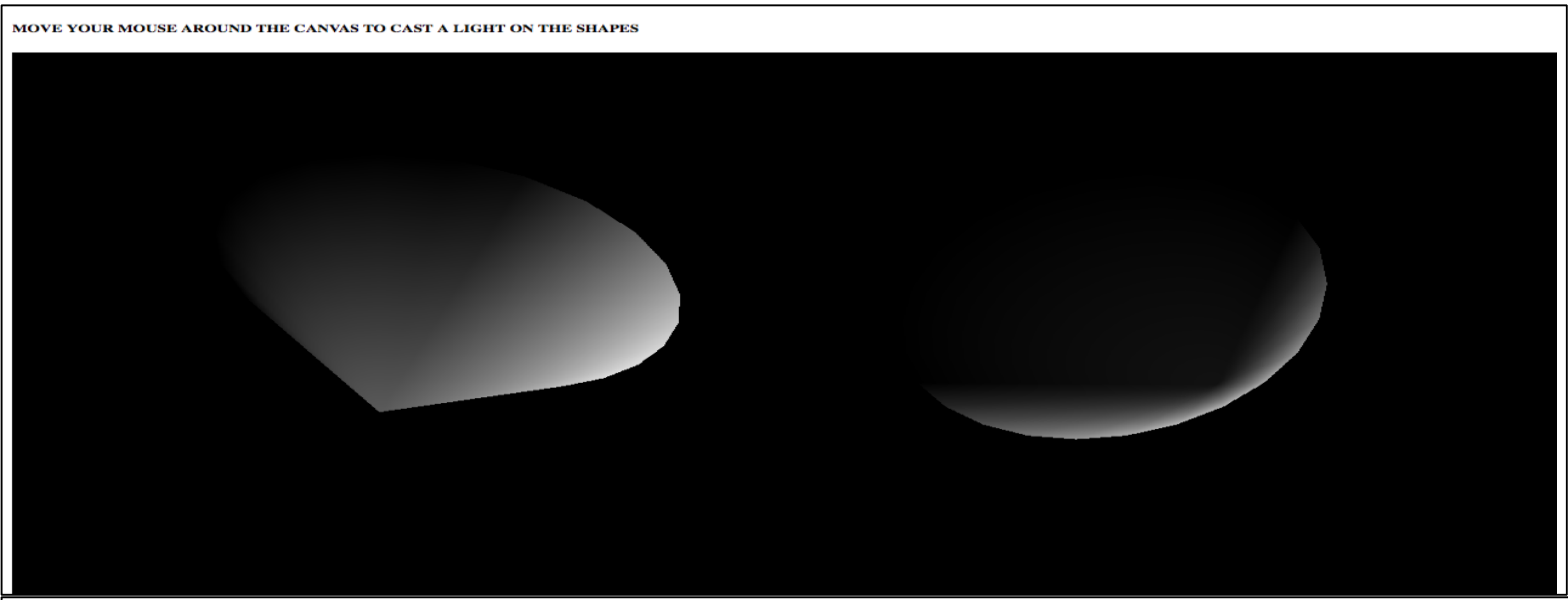
The video submitted along with pdf is a recording of the interface output. Unfortunately, the Quicktime player is only able to record via the internal microphone, so the quality is subpar. The sound grows louder on the left side as that is where the mic is located on my computer. Testing the code out via a live preview on Brackets or Atom would be ideal.

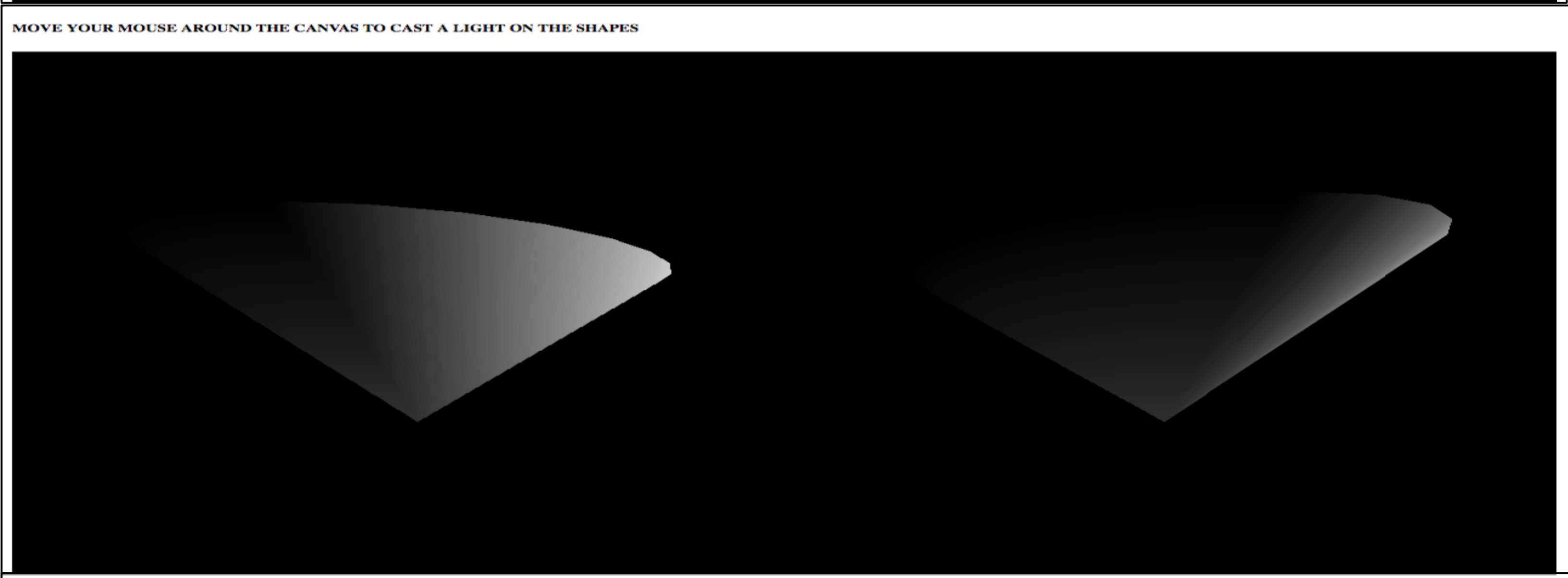
For this sketch, I expanded on my previous sketch – the fireworks – and the pan sound example from the p5 reference page. Using a combination of the two, I created an interface where the mouse dictates the panning of the audio and the display of fireworks. What that means is the mouseX will indicate where the audio will play in the stereo field, whether it be left or right speaker. A mouse click initiates the audio playback and pauses it.

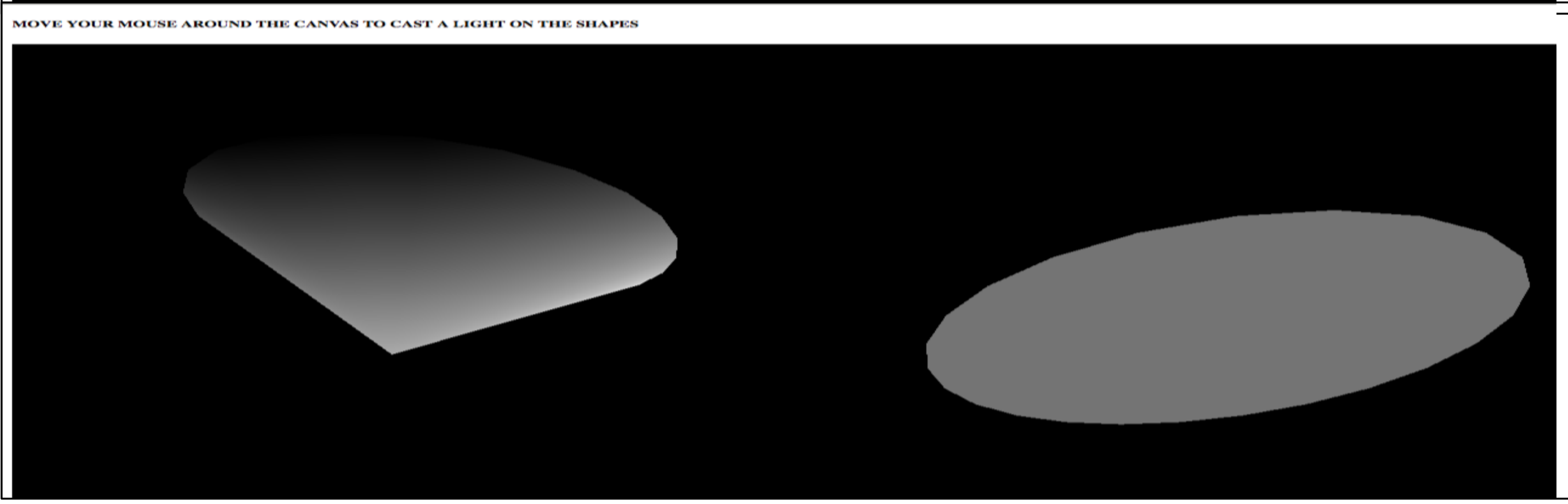
It worked as it was intended to. Given more time I would like to expand on the visual effects rather than resorting to previous code. I would also love to have the mouseY target the depth of stereo field (front and rear). Otherwise, there should also be an option to load a different sound. This could potentially become a codec (if that is the right word) to add to a website as a media player.

Sketch 5

```
// references Directional Lights example
// https://p5js.org/examples/lights-directional.html

// read up on WebGL here
// https://github.com/processing/p5.js/wiki/Getting-started-with-WebGL-in-p5

var radius = 200;

function setup(){
                    createCanvas(window.innerWidth, window.innerHeight, WEBGL);
}

function draw(){
                    noStroke();
                    background(0);
                    var dirY = (mouseY / height - 1) * 5;
                    var dirX = (mouseX / width - 1) * 5;
                    directionalLight(250, 250, 250, dirX, dirY, 1);
                    ambientMaterial(250);
                    translate(-1.5 * radius, 0, 0);
                    rotateX(millis() / 1000);
                    cone(radius);
                    translate(3 * radius, 0, 0);
                    rotateX(millis() / 2000);
                    cone(radius);
}
```


MOVE YOUR MOUSE AROUND THE CANVAS TO CAST A LIGHT ON THE SHAPES


MOVE YOUR MOUSE AROUND THE CANVAS TO CAST A LIGHT ON THE SHAPES


MOVE YOUR MOUSE AROUND THE CANVAS TO CAST A LIGHT ON THE SHAPES

For this sketch, I wanted to exploit the 3D graphics capability in p5. I expanded on the Directional Lights example from the p5 reference page. While utilizing the code I saw the WebGL aspect – which I was unfamiliar with – when creating the canvas. I read about it on a Github tutorial page (linked above) and it essentially allows the canvas to include the z-dimension for 3D rendering. With that in tow I changed the shapes from the initial spheres to cones, as it adds more possibilities for light deflection. Using the rotate function I set each cone to a timer so that they would rotate automatically and at different periods. The mouse indicates where the light is being directed.

I am satisfied with how the sketch turned out, as it did what it was intended to do. Given the chance to expand on it, I would add more rotating possibilities. Perhaps a *mouseDragged()* function could rotate the Y-axis or Z-axis. I would also like the light to be brighter and more predominant. Similar to a flashlight effect. As it is now, you cannot see where the light is concentrated.

Interface Idea

Using these tools, I would like to develop an interface that mimics and recreates the digital rain effect from the Matrix. The Matrix is one of my favourite movies, and the iconography associated with it is the green code falling from above, as shown in the images below. There is even a youtube video with a tutorial. However, I would like to try this on my own as much as possible.

https://www.youtube.com/watch?v=S1TQCi9axzg&t=13s